ADT PLM

Programmer's Learning Machine

Matthieu Nicolas

IJD Seminar, 2016-02-02

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Presentation of PLM

Purposes

- Application to learn programming.

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.

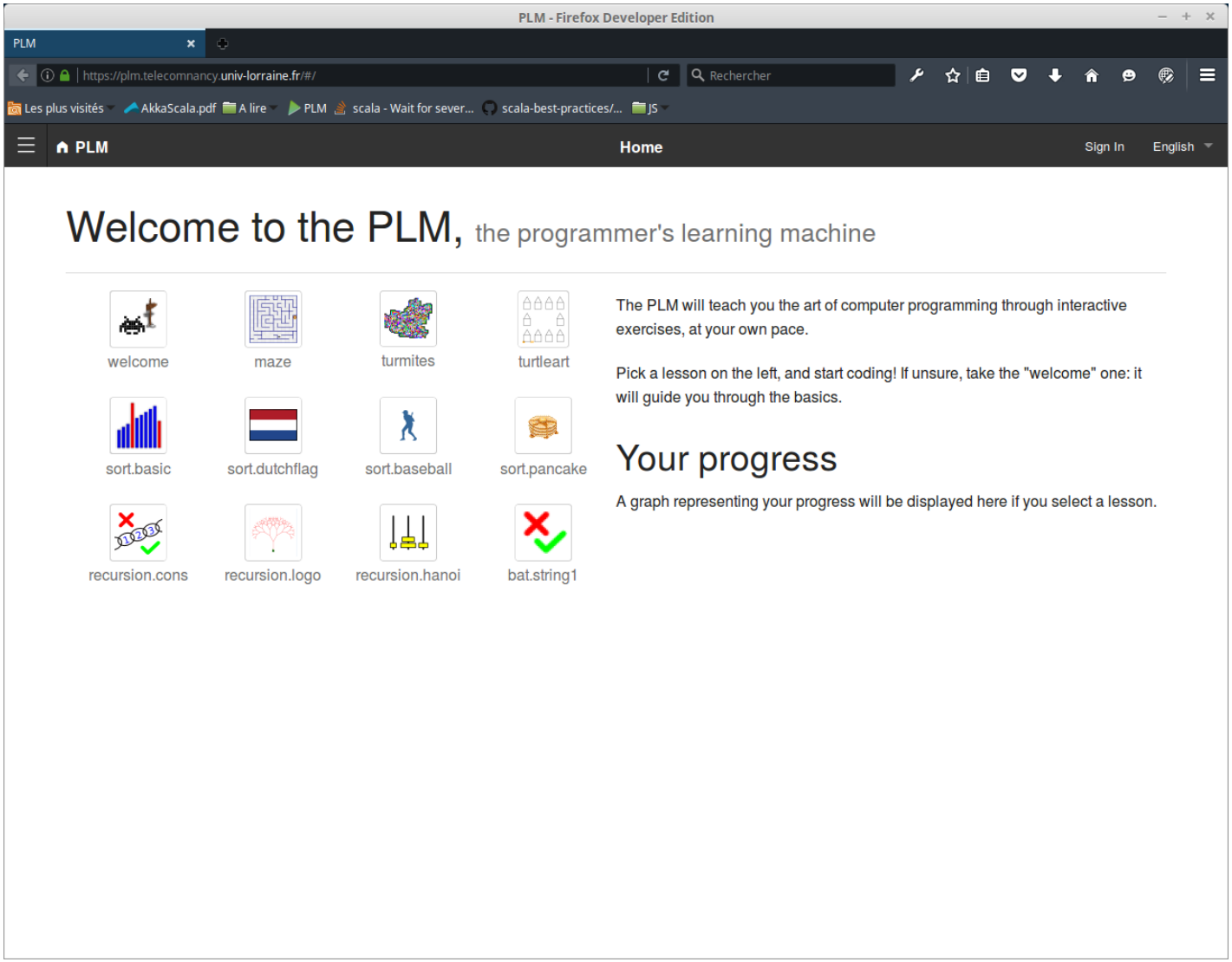
Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.
- Used at TELECOM Nancy since 2008.

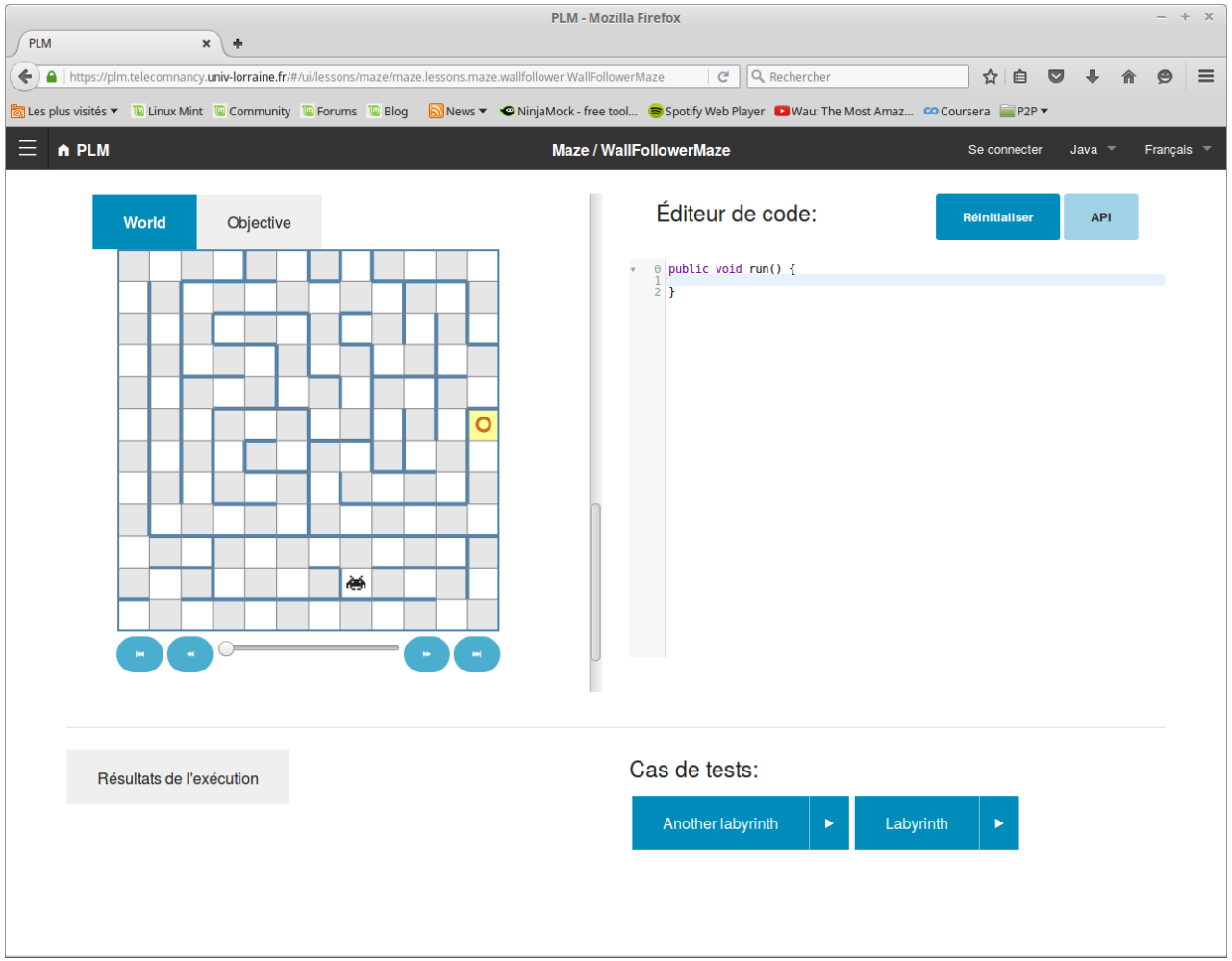
Presentation of PLM

Quick demo



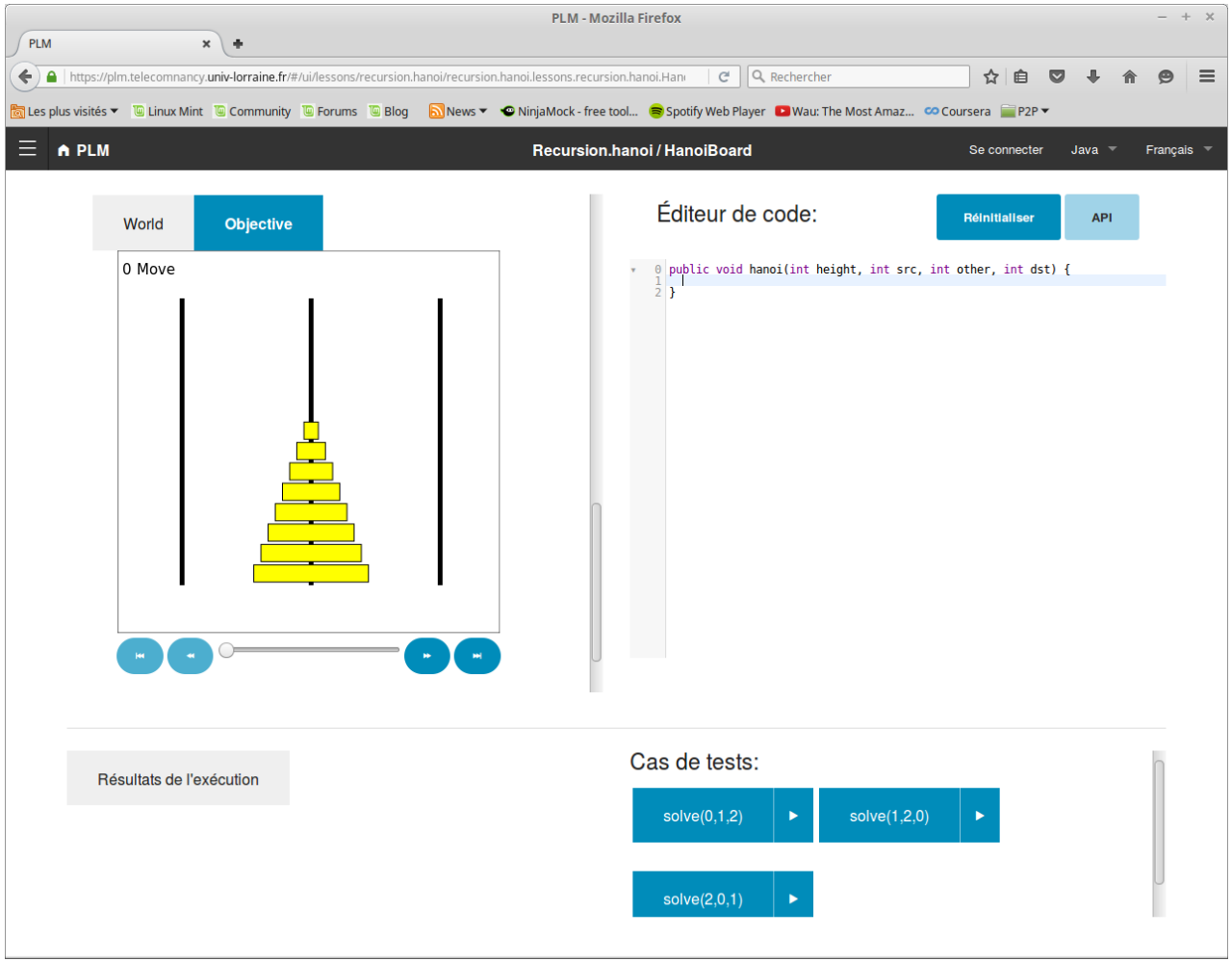
Presentation of PLM

12 lessons, 200 exercises



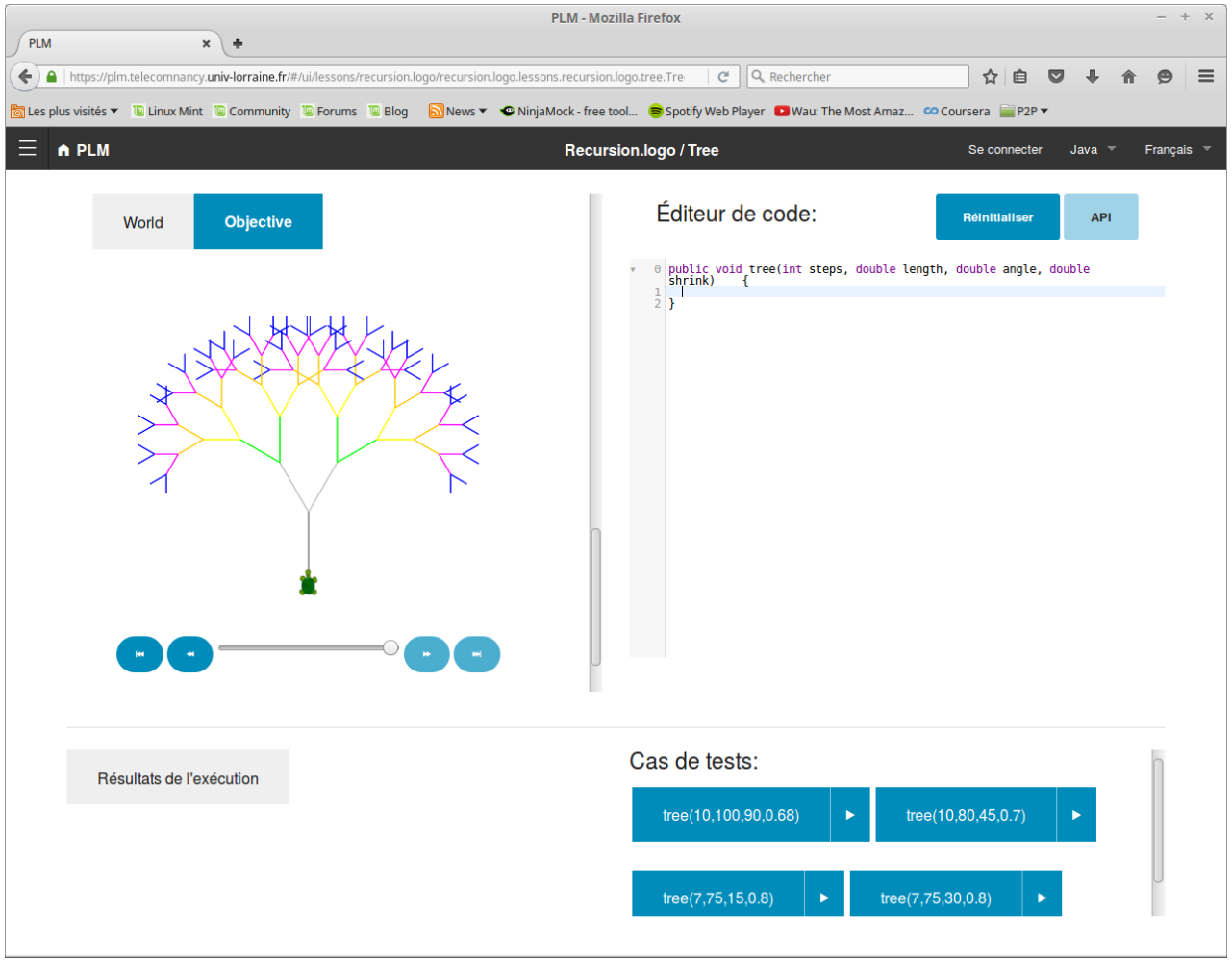
Presentation of PLM

12 lessons, 200 exercises



Presentation of PLM

12 lessons, 200 exercises



Presentation of PLM

Supported languages

- English
- French
- Brazilian Portuguese

Presentation of PLM

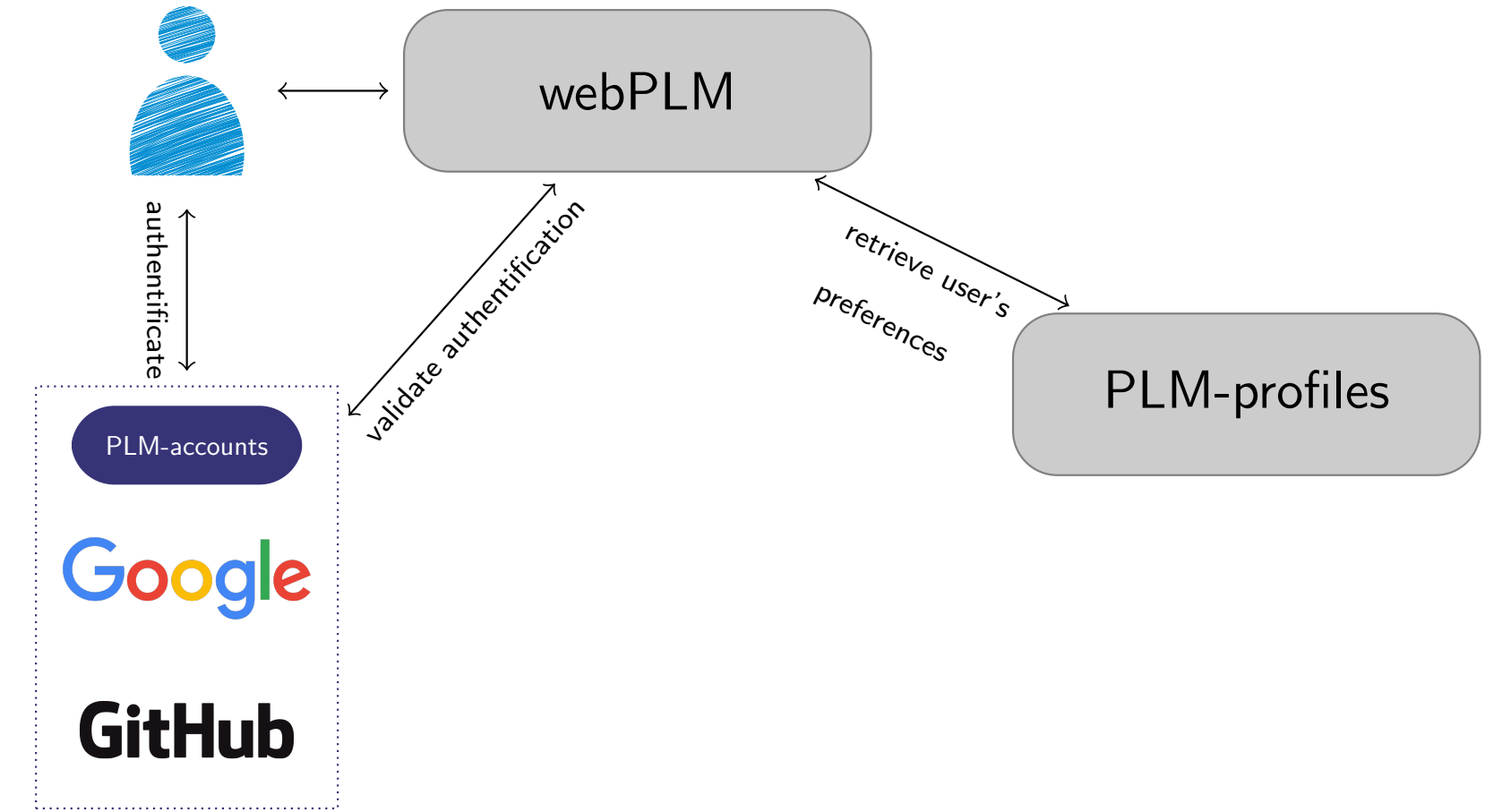
Supported programming languages

- Formerly a fat client
  - Written in Java

- Formerly a fat client
  - Written in Java
- Switch to a web application
  - Server implemented in Scala using PlayFramework
  - User interface written in Javascript using AngularJS and Foundation

Presentation of PLM

Application's architecture

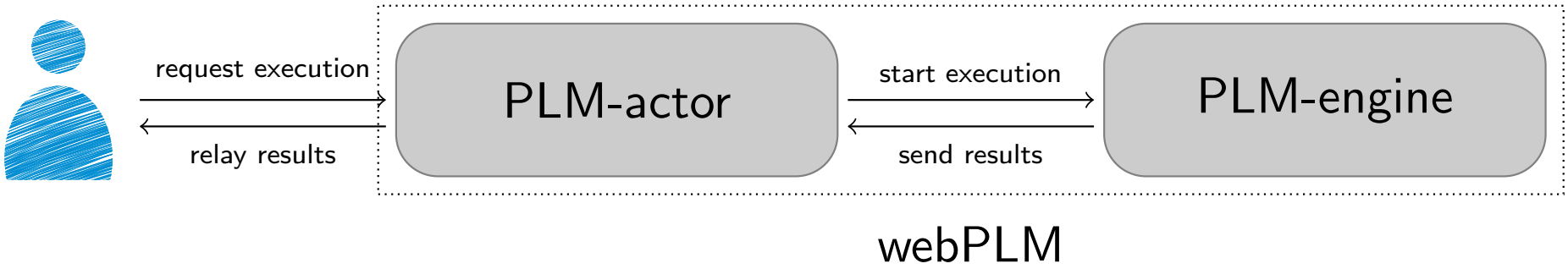


Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

User's code's assessment

Execution components



User's code's assessment

Limits

- Run on the same machine, same JVM

User's code's assessment

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops

User's code's assessment

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Storage jamming with files

User's code's assessment

Limits

- Run on the same machine, same JVM

- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Storage jamming with files
- And from *System.exit(whatever)*?

User's code's assessment

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Storage jamming with files
- And from *System.exit(whatever)*?
- Scalability issues

User's code's assessment

Chosen solution

- Delegate the execution to workers

User's code's assessment

Chosen solution

- Delegate the execution to workers
- *Let it crash* strategy

User's code's assessment

Chosen solution

- Delegate the execution to workers
- *Let it crash* strategy

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements

User's code's assessment

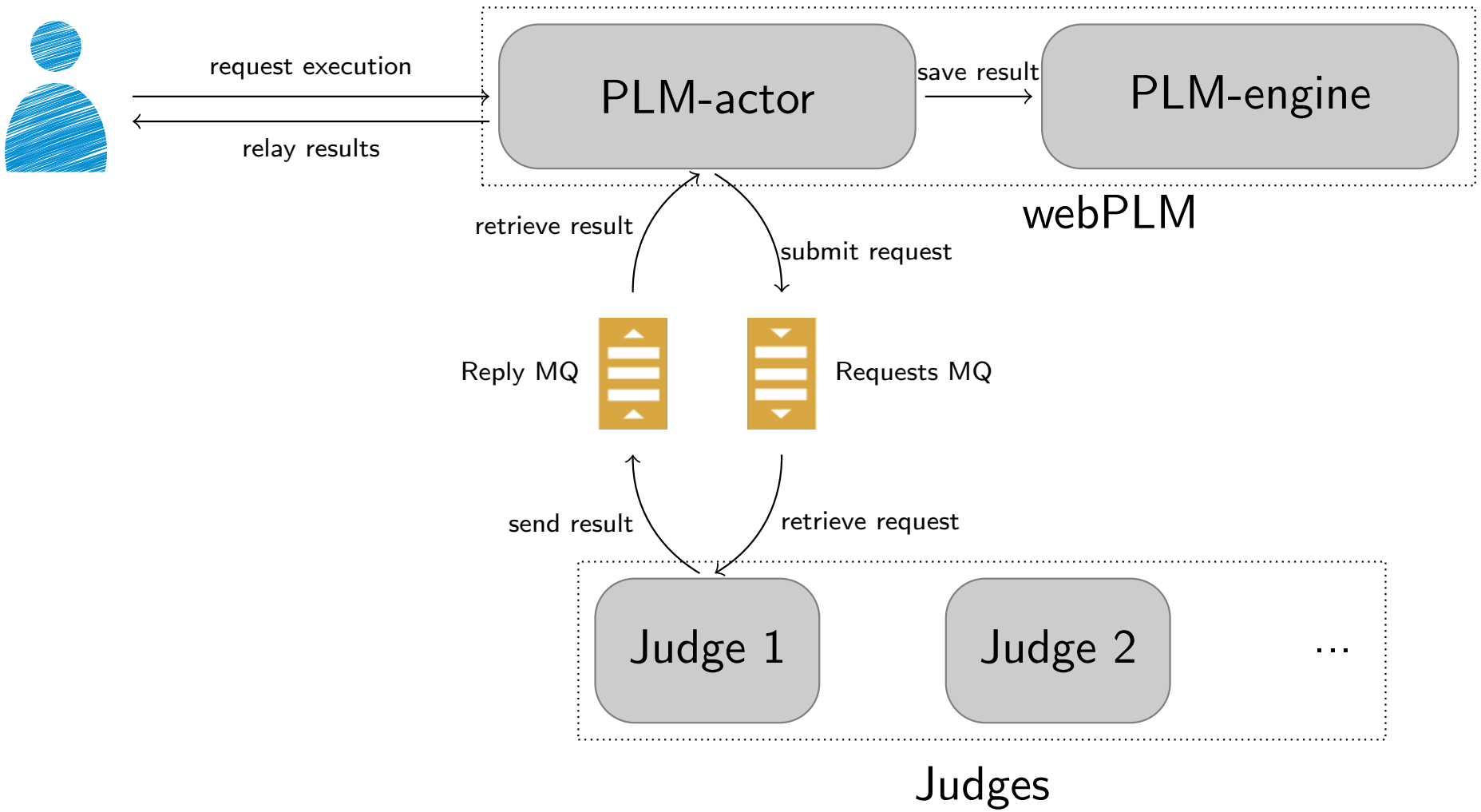
Chosen solution

- Delegate the execution to workers
- *Let it crash* strategy

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements
- Cons:
  - Need to deploy them easily
  - Should be able to reset them
  - Have to restrict their resources usage

User's code's assessment

Architecture with judges



User's code's assessment

Docker

- Build image of your application
- Run containers based on images
- Lightweight virtualization

User's code's assessment

Example of Dockerfile

User's code's assessment

Docker-compose

User's code's assessment

Docker in our case

- Deploy easily all components
- Restart judges automatically
- Hold out against users' mischiefs

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Result

Current architecture

Result

Live-session in TELECOM Nancy

- 30 hours of live testing with 100 students.

Result

Live-session in TELECOM Nancy

- 30 hours of live testing with 100 students.
- Engine is (almost) working fine...

Result

Live-session in TELECOM Nancy

- 30 hours of live testing with 100 students.
- Engine is (almost) working fine...
- ... but user experience needs to be improved!

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.
- No tools for monitoring set up...

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.
- No tools for monitoring set up...
- ... so the bottleneck is unknown.

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Next steps

Refactor the code

- Rushed to release a stable version before September...
- Needed to refactor some parts of the code.
- Standardized behavior of local and server mode.

Next steps

Simplify workflow to adapt the content

- Store most content inside PLM.
- Heavy and error prone workflow.
- Need to extract the content from PLM's jar.
- Allow to implement an exercise editor.

Next steps

Solve performance issues

- Set up some monitoring tools.
- Perform some load testing to identify the bottleneck.

Questions

Thanks for your attention, any questions?